

Getting started with Oracle in the Amazon cloud

By Guy Harrison

Cloud computing is a heavily overused term, applied to wide range of offerings such as Gmail and Flickr as well as Cloud platforms such as Windows Azure, Google App Engine and Amazon Web Services environment. I generally define cloud computing as *the provision of virtualized application software, platforms or infrastructure across the network, in particular the internet.*

Web Services (AWS) have taken an early market, technology and mindshare the lead in the Infrastructure as a Service (IaaS) category of cloud computing. AWS and similar offerings provide virtualized server resources together with other infrastructure (messaging, database, etc) over the internet. Customers build their application stack using these components. Amazon probably provides the richest IaaS stack including distributed storage (S3), cloud database (SimpleDB), messaging (SQS) and payments (FPS).

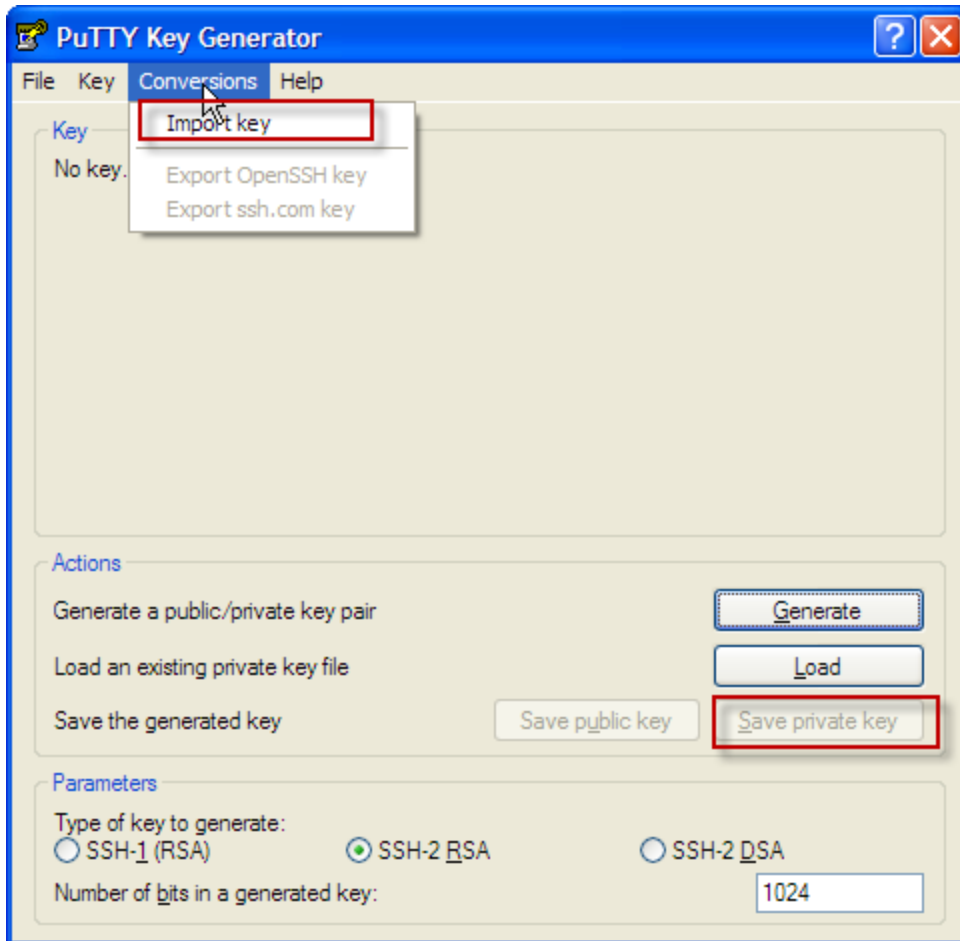
Ever since Oracle announced support for Amazon AWS at Oracle Open World I've been eager to play around with it. This week I managed to find the time to set up a test database and thought I'd share my experiences.

Getting into AWS

Getting started with Amazon AWS is a big topic. In a nutshell:

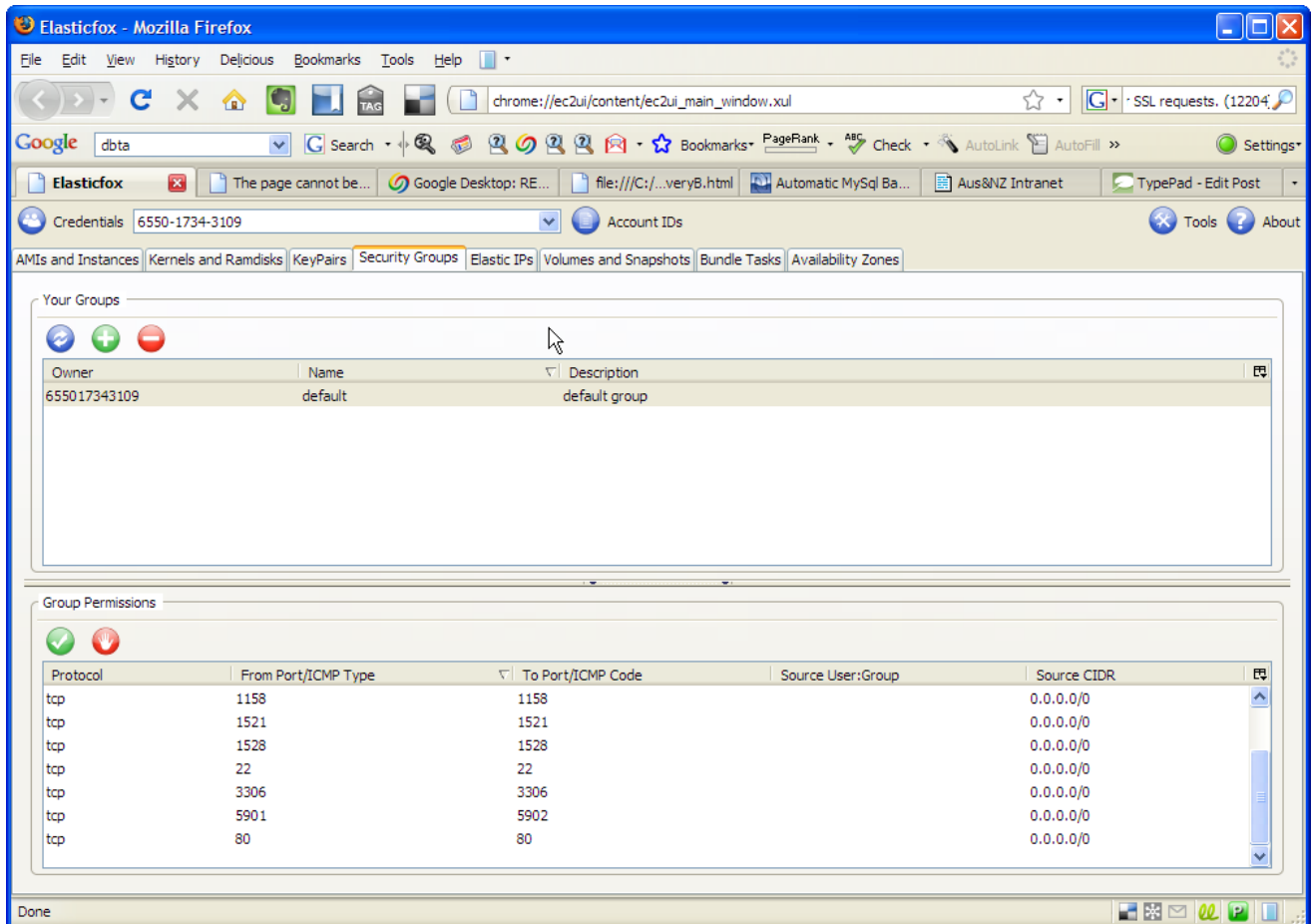
1. Get an AWS account from www.amazon.com/aws (you need your credit card)
2. Download and install ElasticFox (<http://sourceforge.net/projects/elasticfox/>). I've used the Amazon command line utilities in the past, but ElasticFox is far easier. Another alternative is to use the free web based dashboards at Rightscale.
3. Install putty (www.putty.org) - a free SSH client
4. Enter your AWS credentials in ElasticFox
5. Create a keypair in the keypairs tab of ElasticFox

Most of the above went smoothly for me, other than the use of putty in ElasticFox. It turns out that Putty keypairs are not compatible with normal AWS keypairs so you have to create a putty compatible key. To do this, import your .PEM file in the PUTTYGEN program that comes with putty and generate a .PPK key which you then use when connecting via Putty.



Port configuration

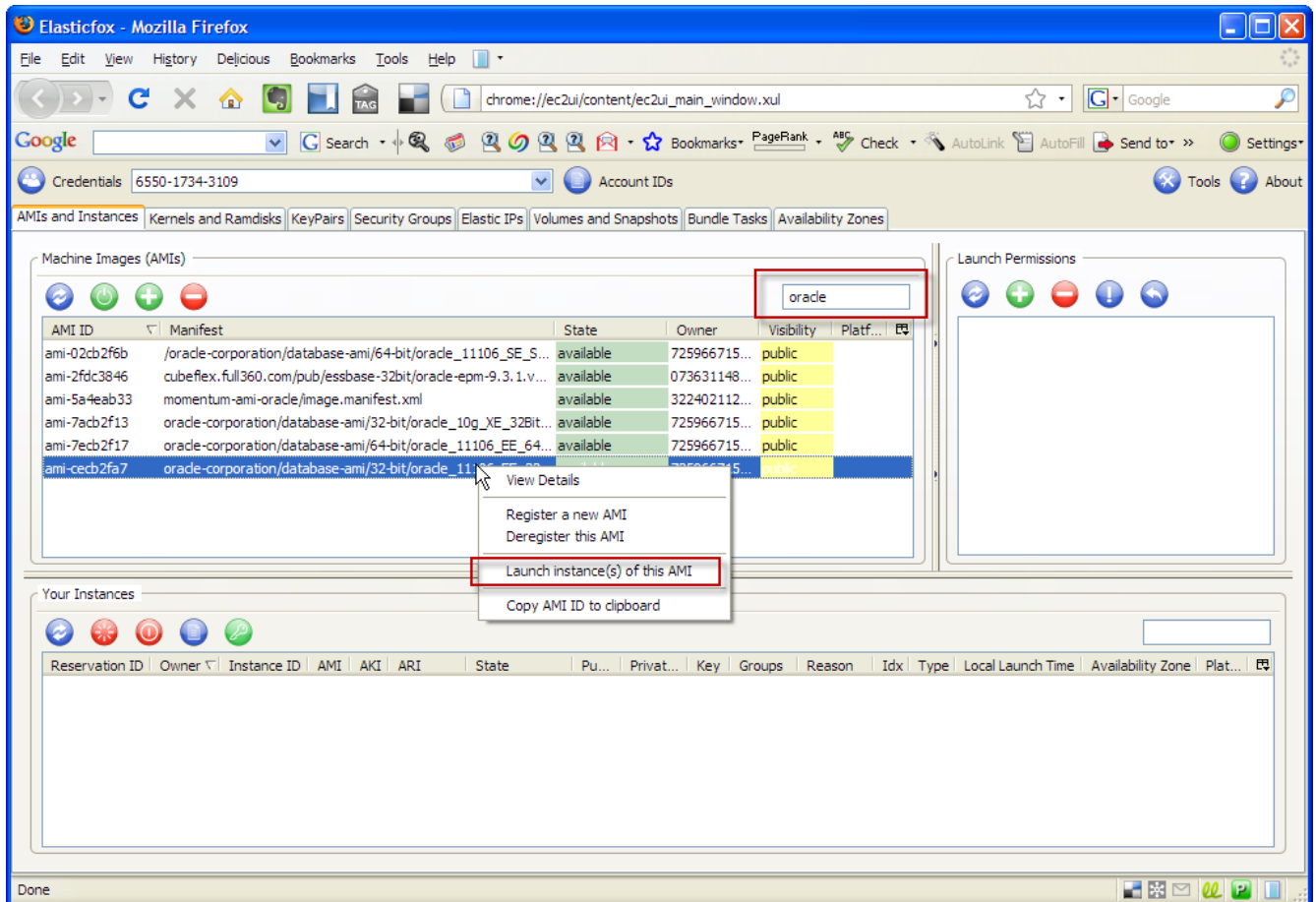
We need to open various ports to allow us to use SQL*NET, OEM etc. ElasticFox lets you change the default port settings in the security groups page:



I've got 3306 open for MySQL, and 5901-5902 so I can run VNC servers in the EC2 instance. 1521 is for SQL*NET of course, and 1158 is for enterprise manager. 22 and 80 are standard for SSH and HTTP.

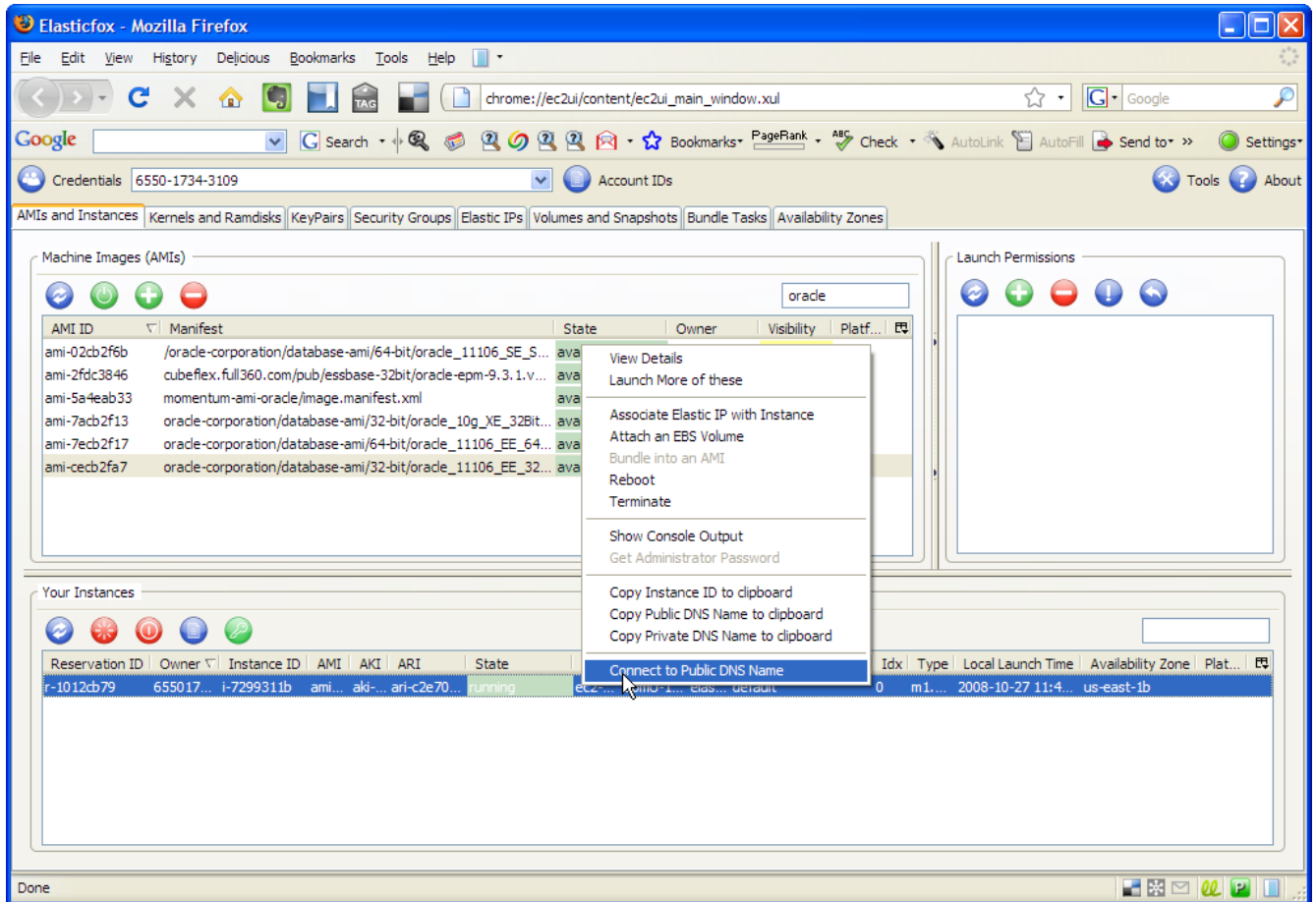
Starting the EC2 instance

Amazon Machine Images (AMIs) are template servers that you can start with. It's easy enough to find the oracle AMIs, just specify "oracle" in the ElasticFox search box. Then you can select the one you want (in this case 32-bit Oracle 11) and select "Launch instance(s) of the AMI":



You need to wait for the new instance to start. About 5 minutes should do it (about the time it takes to startup a linux host on physical hardware).

Once the instance is running we can connect to it by right clicking and selecting "connect to public DNS name". This is where things will go wrong if you haven't created a PUTTY compatible private key file. If everything is set up correctly you will get a SSH session on your new EC2 instance (you should not be prompted for a password).



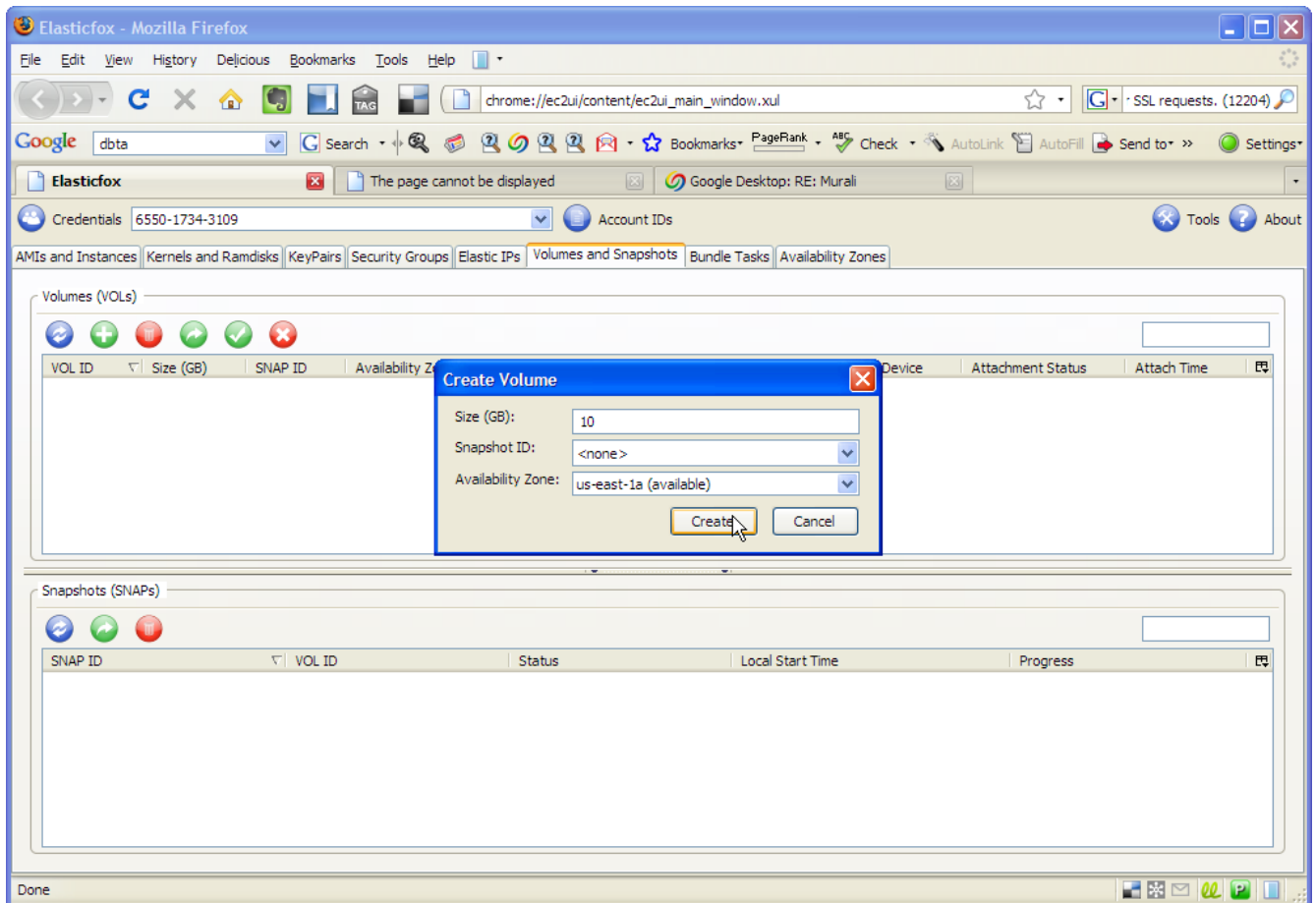
All being well, you will be asked to accept the Oracle licensing agreement and then it will ask you if you want to create a database. Say "no" for now, since we are going to create our database on Elastic Block Storage (EBS).

Elastic Block Storage

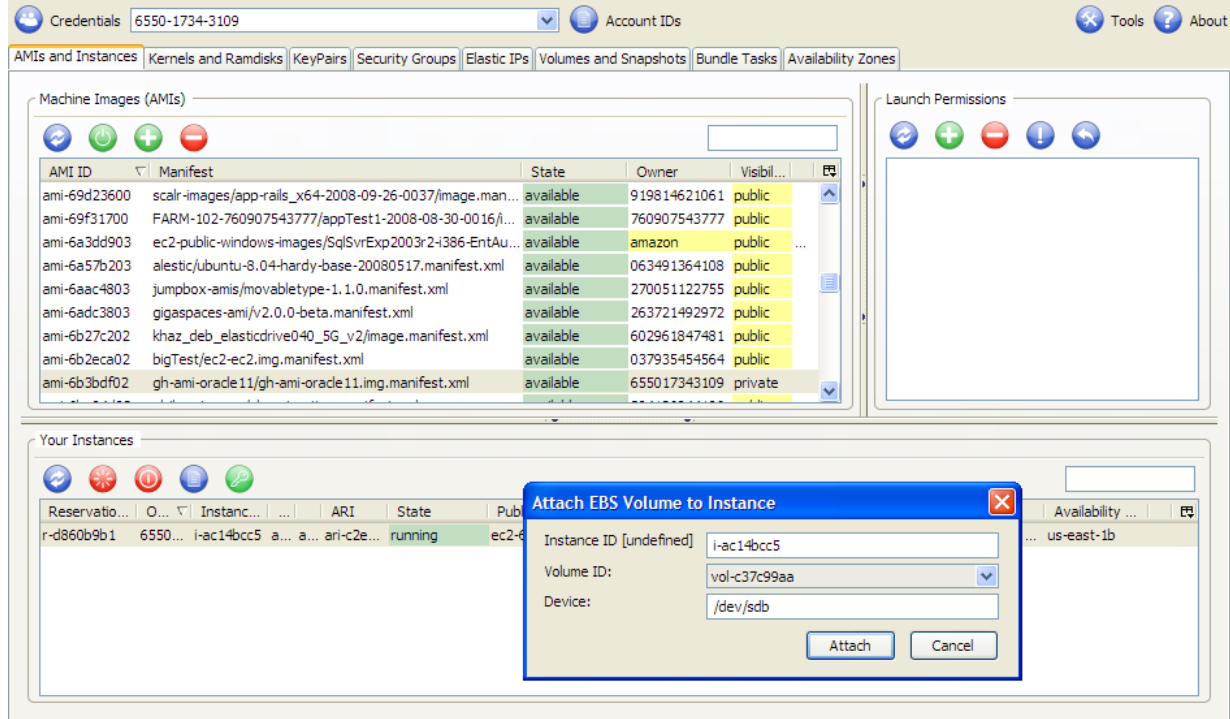
An EC2 instance is a transient virtual machine image. Any changes you make to the virtual machine will be lost when you terminate the instance. Obviously we don't want our database to just disappear, so we need a permanent solution.

Firstly, we want our database files on persistent storage. That means we need an Elastic Block Storage (EBS) device. EBS appear like raw partitions on the EC2 host, but they have a separate and persistent existence.

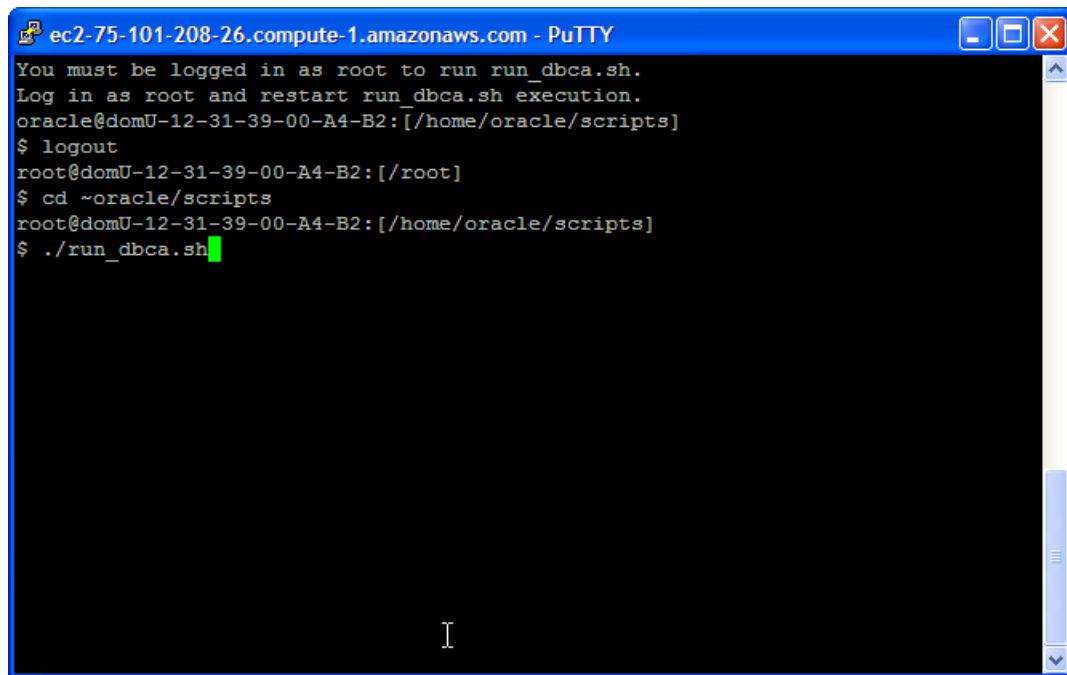
EBS volumes can be created in ElasticFox in the "Volumes and Snapshots" page. Below I'm creating a 10GB volume where I will put the database files. Make sure that the volumes are in the same availability zone as your instance. In the screen shot below I created the volume in 'us-east-1a', but my EC2 instance is in 'us-east-1b' (see screen shot above). Consequently I had to discard and recreate this volume before I could use it.



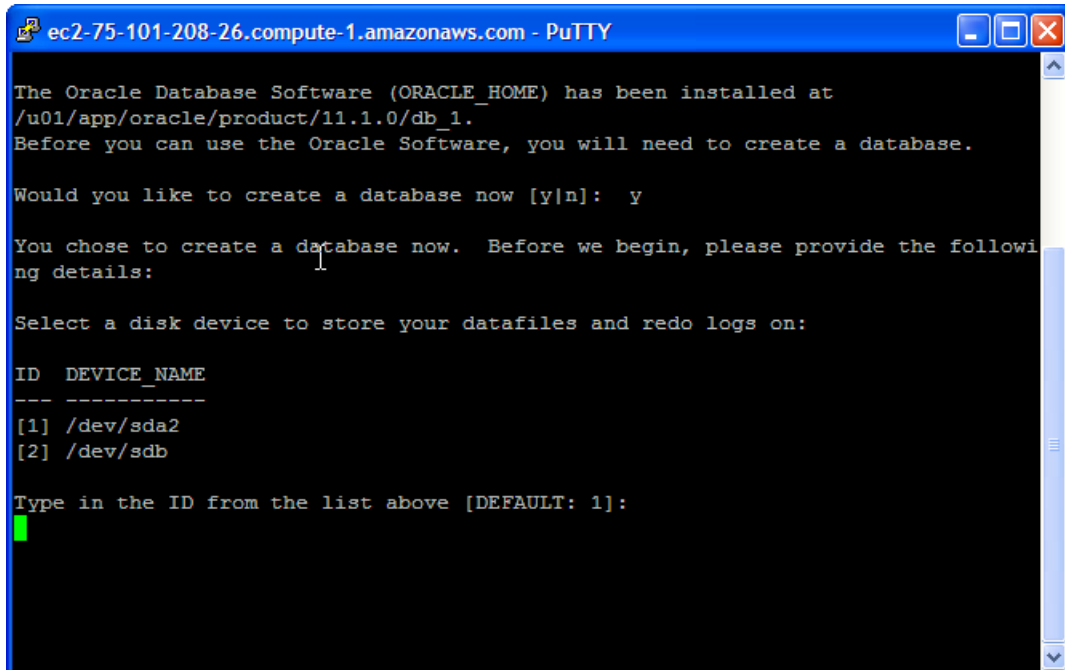
Now, attach the EBS volume to your instance. Assign a device such as '/dev/sdb':



Now we are ready to create a database. Below I cd to the ~oracle/scripts directory (as root!) and run the run_dbca.sh script:



Creating the database is straight forward, but when asked where you want to create the database, make sure you specify /dev/sdb (or whatever device you used to mount your EBS volumes):

A screenshot of a PuTTY terminal window titled "ec2-75-101-208-26.compute-1.amazonaws.com - PuTTY". The terminal displays the following text:

```
The Oracle Database Software (ORACLE_HOME) has been installed at
/u01/app/oracle/product/11.1.0/db_1.
Before you can use the Oracle Software, you will need to create a database.

Would you like to create a database now [y|n]: y

You chose to create a database now. Before we begin, please provide the following
details:

Select a disk device to store your datafiles and redo logs on:

ID  DEVICE_NAME
---  -
[1] /dev/sda2
[2] /dev/sdb

Type in the ID from the list above [DEFAULT: 1]:
```

If you create your database on /dev/sda2, then the database files will be lost when the instance is terminated. Because /dev/sdb is on an EBS storage volume, it will persist even if the EC2 instance goes down. So our database files will not be lost.

Creating an AMI template

Now we have a database on our EC2 instance, and the database files are on permanent storage. But if the EC2 instance goes down I will have lost whatever customizations I've made (such as installing perl DBI, setting up VNC servers, etc) **and** I'll have to somehow get the datafiles attached to a new database on a fresh image. What I really want to do is save a copy of my EC2 image when I've got it the way I want it.

The way to do this is to create an Amazon Machine Image (AMI) template. Here are the steps:

1. Shutdown the database
2. Use ec2-bundle-vol on the EC2 image to create a bundle containing the image definition.
3. Use ec2-upload-bundle to copy the bundle into S3 (Simple Storage System) files.
4. Use ec2-register to assign the S3 files to an AMI image

You need more space than usual to create an image of the Oracle system. The maximum bundle size is 10G and I ran out of space initially when trying to create such a large bundle. So I created and mounted a 20GB EBS volume and mounted it as /bundle:

```
$ df -k
Filesystem            1K-blocks      Used Available Use% Mounted on
/dev/sda1             10321208    8352536   1444384   86% /
none                  870484      521772    348712   60% /dev/shm
/dev/sdb              10321208    3602004   6194916   37% /oradata
/dev/sdc              20642428    9963332   9630520   51% /bundle
```

I created the bundle with the following commands:

```
ec2-bundle-vol --prefix gh-ami-oracle11.img \
  --destination /bundle \
  --exclude /oradata,/bundle,/mnt \
  --cert /mnt/cert-N7X7OBMXFZVODIRJOZX3ZIVHTDO6F7LL.pem \
  --privatekey /mnt/pk-N7X7OBMXFZVODIRJOZX3ZIVHTDO6F7LL.pem \
  --user XXXXXXXXXXXX \
  --arch i386
```

The --user option provides your AWS account number without any hyphens. The --cert and --privatekey arguments require your digital certificate and private key files that should have been created for you when you first signed up for AWS.

Next I upload the bundle to an S3 bucket

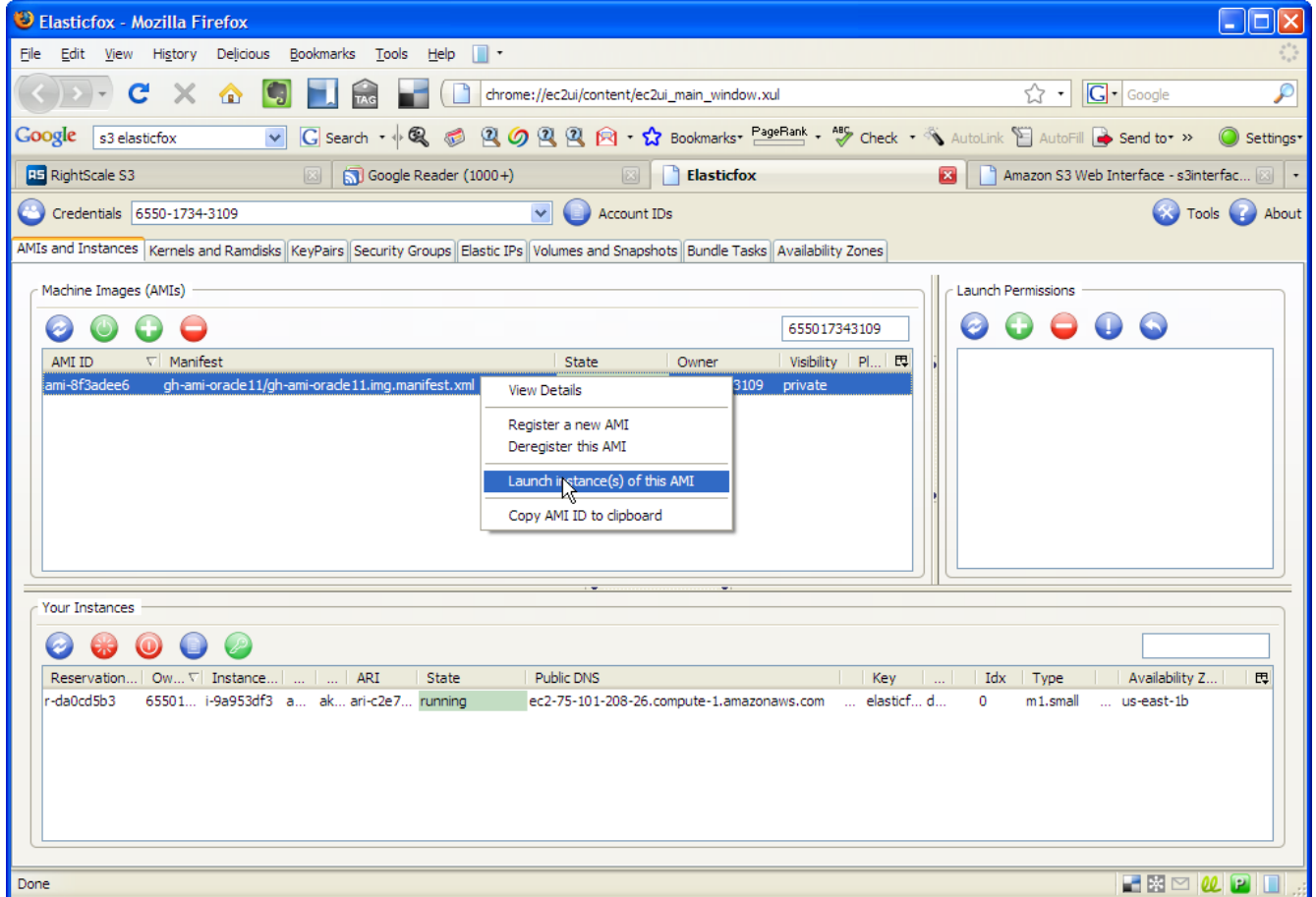
```
ec2-upload-bundle \
  --manifest /bundle/gh-ami-oracle11.img.manifest.xml \
  --bucket gh-ami-oracle11 \
  --access-key `cat aws_access_key.txt` \
  --secret-key `cat aws_secret_key.txt`
```

The access-key and secret-key arguments expect you to pass the keys on the command line, but it's probably safer to cat them from files as shown above. Also I don't want to give you my keys! ☺.

Finally, I register the bundle as an AMI image. The ec-register command is part of the AWS command line toolkit, which I have installed on my desktop:

```
bash-3.2$ ec2-register gh-ami-oracle11/gh-ami-oracle11.img.manifest.xml
IMAGE    ami-8f3adee6
```

My AMI template is now visible from ElasticFox, ready for me to launch:



Setting up a new server

So now my configuration is saved to the AMI template and the datafiles are safely stored in my EBS volume. If I lose my existing server I can "simply" start up a new version of the EC2 instance from the template and attach the datafiles. Well, maybe "simply" is an exaggeration!

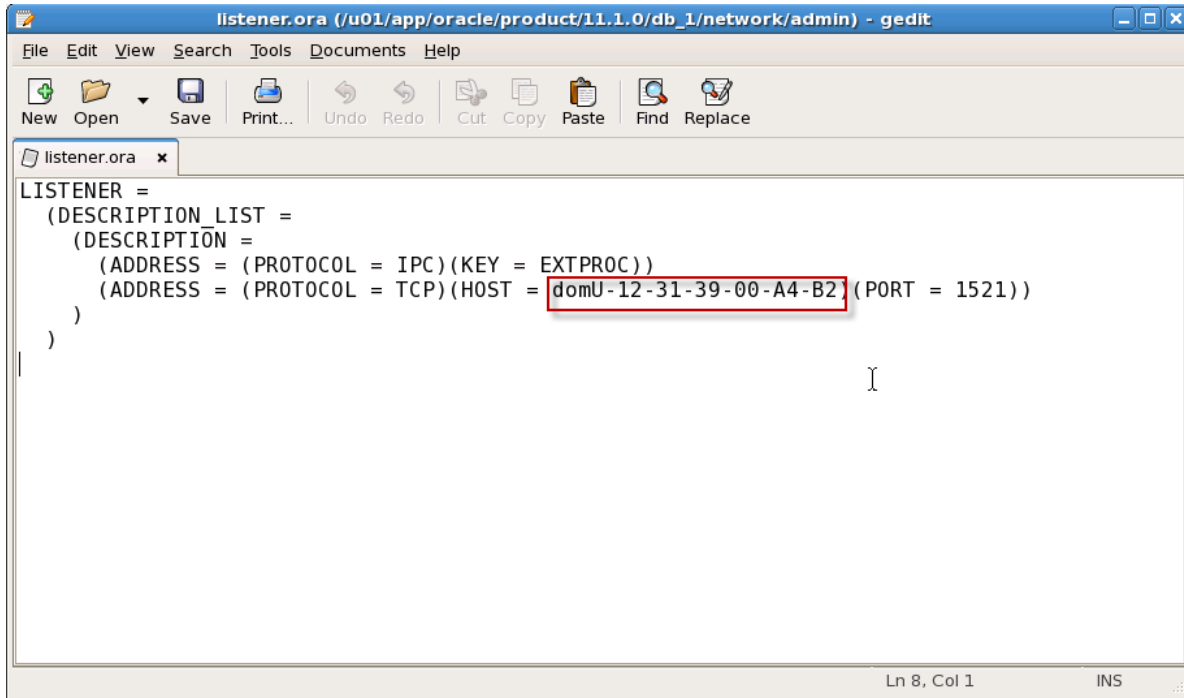
There are two main things that have to happen before I can restart my database on a new image:

1. I need to attach the EBS storage to the new EC2 server
2. I need to adjust the Oracle listener and OEM to the new host name.

When I launch my new instance, it won't boot up all the way unless the EBS volumes that were mounted when I bundled it are present. There is an option to provide a different fstab file with your bundle, but for now I need to attach my EBS volumes to the instance and then reboot it.

The second issue is a bit fiddlier. The new EC2 instance has a new hostname, and the listener and the OEM dbconsole are associated with the old instance name.

Firstly, we should change the listener.ora file so that the hostname is the private DNS name (you can copy this to the clipboard from ElasticFox):



```
listener.ora (/u01/app/oracle/product/11.1.0/db_1/network/admin) - gedit
File Edit View Search Tools Documents Help
New Open Save Print... Undo Redo Cut Copy Paste Find Replace
listener.ora x
LISTENER =
  (DESCRIPTION_LIST =
    (DESCRIPTION =
      (ADDRESS = (PROTOCOL = IPC)(KEY = EXTPROC))
      (ADDRESS = (PROTOCOL = TCP)(HOST = domU-12-31-39-00-A4-B2)(PORT = 1521))
    )
  )
Ln 8, Col 1 INS
```

After you've edited the LISTENER.ORA file you should restart the listener.

You also need to change the value of ORACLE_HOSTNAME in `~oracle/.bash_profile`. In this case you want to use the public DNS name. Again, you can cut this to the clipboard from the ElasticFox screen.

Finally, rebuild Enterprise Manager configuration by issuing the following commands from the Oracle account (in this example, the passwords for all the relevant accounts are held in the `$PASSWORD` variable):

```
emca -deconfig dbcontrol db -repos drop -SID ghec2a -PORT 1521 \
    -SYSMAN_PWD $PASSWORD -SYS_PWD $PASSWORD
emca -config dbcontrol db -repos create -SID ghec2a -PORT 1521 \
    -SYSMAN_PWD $PASSWORD -SYS_PWD $PASSWORD -DBSNMP_PWD
$PASSWORD
```

All done! Listener, database and OEM dbconsole should now be running successfully on the new EC2 host.

A quick look in Spotlight

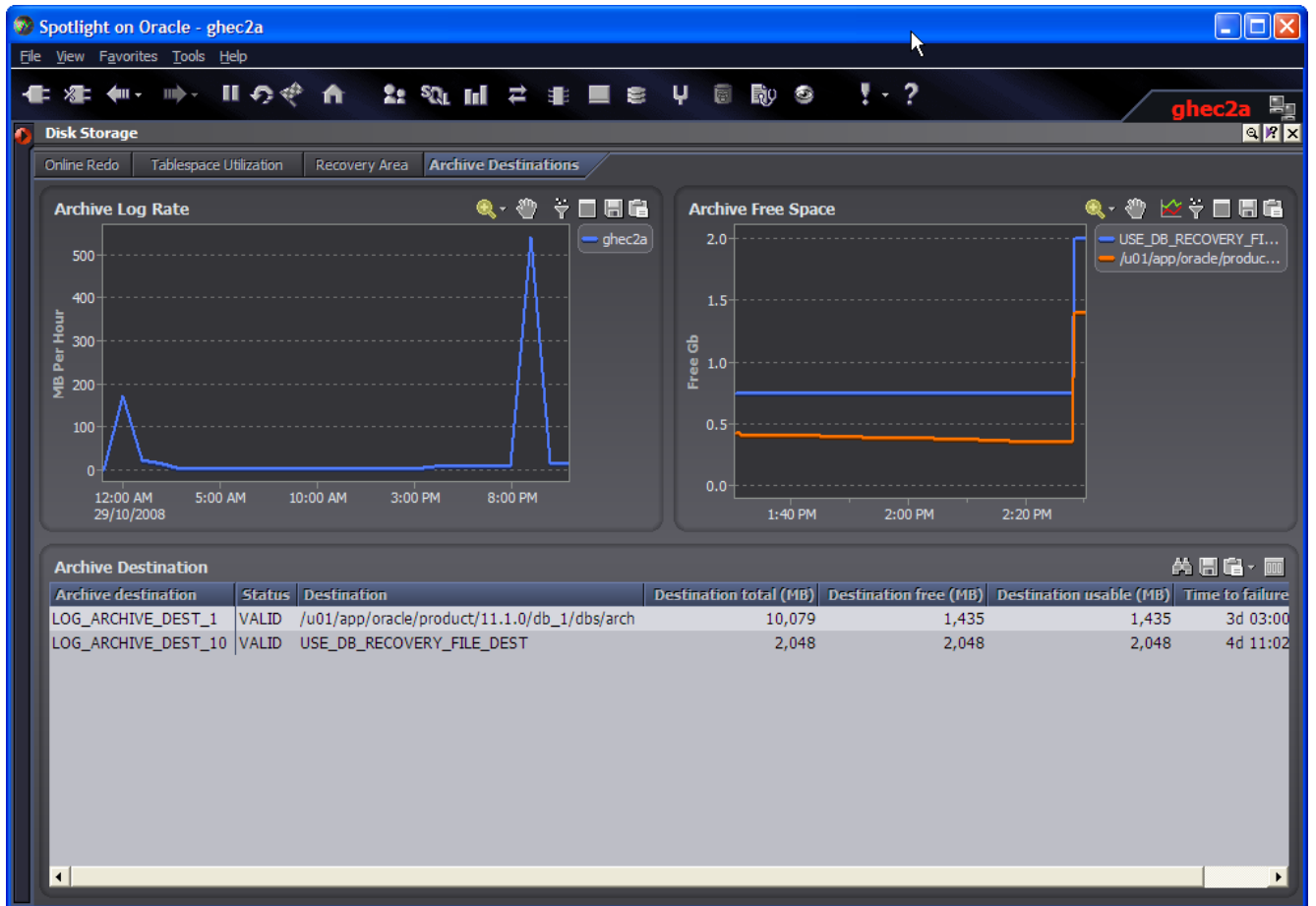
I'll be using Amazon images from time to time in the future for research purposes, but the first thing I was curious about was to see how the database looked in Spotlight. So here's the Spotlight home page for the database I just set up:



Whoops! The original Oracle image had archivelog on and now I'm almost out of space. It's archiving both to the Recovery area and to \$ORACLE_HOME/dbs/arch. The recovery area is on the EBS volume, but the filesystem location is within the Oracle Home and therefore on my EC2 instance storage. In the future, I'll turn off archiving before creating an AMI, since not only could I potentially run out of disk, but also the AMI is bigger than it should be since it has a couple of 100MB of archive logs. I cleared the logs using RMAN:

```
connect target sys/mypassword
CONFIGURE DEVICE TYPE DISK BACKUP TYPE TO BACKUPSET;
CONFIGURE CHANNEL DEVICE TYPE DISK FORMAT
'/oradata/tmp/ora_df%t_s%s_s%p';
BACKUP ARCHIVELOG ALL DELETE ALL INPUT;
```

Now when I look in the Spotlight archive destinations drilldown the situation is OK, and I have a few days of archivelog left (at current allocation rates):



Note that the archive management screen above is part of a pre-release version of Spotlight that will be generally available next month.

Conclusion

I'm a big believer in the style of cloud computing pioneered by Amazon and I'm really pleased to see Oracle playing a part in the Amazon cloud. It's pretty easy to get started with Oracle in AWS - especially if you have a basic familiarity with the AWS environment - but a bit harder to establish a persistent database environment. For the Open Source Software stacks that dominate within AWS, companies such as [Rightscale](#) offer facilities to help automate the constructions and deployment of solutions.

Hopefully there'll be similar facilities for Oracle in AWS soon. In the meantime, I hope that sharing my experiences helps in some small way.

About the Author

Guy Harrison is chief architect, database solutions, at Quest Software, Inc. He is a recognized database expert with over 15 years of experience in database administration, development and performance tuning. Guy is the author of *Oracle SQL High Performance Tuning* (Prentice-Hall, 1997, 2000), *Oracle Desk Reference* (Prentice-Hall, 2000) and *MySQL Stored Procedure Programming* (O'Reilly, 2006) and is a regular contributor to database-related technical journals and a regular speaker at technical conferences. Guy was instrumental in creating Quest's popular Spotlight[®] product line, and contributed to the development of other Quest products such as Quest Central[™] and Foglight[®].